Course Objective:
Testing and verification of the software systems is not a “silver bullet” that can guarantee the production of high quality applications. While a “correct” correctness proof demonstrates that a software system (which exactly meets its specification) will always operate in a given manner, software testing that is not fully exhaustive can only suggest the presence of flaws and cannot prove their absence. It has noted by various researchers and industry professionals that it is impossible to exhaustively (or completely) test an application because: (1) the domain of program inputs is too large, (2) there are too many possible input paths, and (3) design and specification issues are difficult to test. The first and second points present obvious complications and the final point highlights the difficulty of determining if the specification of a problem solution and the design of its implementation are also correct.

There is no denying that IT has entered into each one of our lives. Invariably any device we use now days such as TV, mobile phone, car, etc., has some kind of information processing unit in it. It is important that the software running in these devices should be functionally correct. A software error in a critical system such as nuclear reactor could however be catastrophic. Software systems are therefore always checked for correctness before they are deployed. Verification and testing are two predominant ways for checking the correctness of software systems. This course is aimed at learning various techniques of testing and verification.

Course Contents:

Lectures
Module 1: Software Testing: Basic concepts and preliminaries
Module 2: Model-based Testing
Module 3: Design and Code Inspections to Reduce Errors in Program Development
Module 4: Applications of AI in Software Quality Assurance
Module 5: Verification of Software Systems
Module 6: Testing of Web GUI and Mobile Apps
Module 7: AI and ML for Security Testing

Lab Sessions
The lab sessions of the course use open source testing tools (e.g., JUnit Unit Testing Framework, TestNG Integration Testing Framework, MuJava (Mutation Testing), Static Analysis tools – Findbugs, CodePro, PMD, NFR testing tools, GUI testing tools) and handouts/exercises to generate effective test cases from the source code (or software project artifacts)

Grading Policy:
- Quizzes - 15%
- Lab/Assignment/Project and Viva - 25%
- In-Semester Exam - 25%
- Final Exam - 35%

Online Delivery of Lecture and Labs:
The class size is not too large (25 students), hence, I have planned to use Google Meet/Zoom for taking the online lectures/labs. The earlier experience with Google Meet is satisfactory. For the lab sessions, I will use only open source tools which students can download and install in their machines. For MIS, Google classroom will be used.